Model selection and plotting performance

Nova F. Smedley

2018-11-04

The package `gbmSPM` can be used to do the exact analysis pipeline used in the paper,

Smedley, Nova F., Benjamin M. Ellingson, Timothy F. Cloughesy, and William Hsu. "Longitudinal Patterns in Clinical and Imaging Measurements Predict Residual Survival in Glioblastoma Patients." *Scientific reports* 8, no. 1 (2018): 14429.

See also Supplemental Materials.

Vignette Info

This vignette shows how temporal features and patient covariates are used in predicting residual survival. This uses dummy patient data and follows the other vignettes "Generate sequential patterns" and "Predicting residual survival."

Example

1. **Create a few types of temporal features**

   To use dummy data, run `example_spm.R` from the `examples` folder in the terminal:

   ```
   $ Rscript /PATHTO/example_spm.R --tType 'rate' --minSupp 0.2 --minSuppList 'yes' --maxgap 60 --maxl
   ```

   Optionally: repeat the above for `tType` of `p` for percent change instead of rate change in tumor volume measurements:

   ```
   $ Rscript /PATHTO/example_spm.R --tType 'p' --minSupp 0.2 --minSuppList 'yes' --maxgap 60 --maxleng
   ```

   This will generate patterns and feature vectors stored in `gbm_spm_example_multi`.

2. **Use the generated features in predicting residual survival**

   To use dummy data, run `example_logit_spm.R` from the `examples` folder for the different SPM parameters:

   ```
   $ Rscript example_logit_spm.R --tType 'rate' --maxgap 60 --maxlength 2 --lmax 0.2 --llength 100 --o
   ```

   Optionally: again, repeat the above for the different SPM patterns created:

   - `sup0.2g60l2z2`
   - `sup0.25g60l2z2`
   - `sup0.3g60l2z2`
   - `sup0.35g60l2z2`
   - `sup0.4g60l2z2`

   and/or also repeat for another `tType` of `p` for percent change.

   This will perform repeated 2-fold cross-validation for hyperparameter selection. Experiment logs and logit model (and results) are also stored in `gbm_spm_example_multi`.

3. **Load functions to format logit results**

   These are functions for processing the data, which were stored and named in a specific manner that would likely be different than how users may use `gbmSPM`.

```
library(gbmSpm)
library(grid)
library(scales)
library(wesanderson)

getCVResults <- function(dir, fn, name) {
  cvResultsFile <- file.path(dir,fn)
  cvResults <- read.table(cvResultsFile, header=T, sep=',')

  # identify the cspade parameters the logit's features were based on
  temp <- do.call(rbind, lapply(cvResults$model, function(n) regmatches(n, gregexpr('[0-9]+',n))[[1
  colnames(temp) <- c('support', 'gap', 'length', 'size')
  temp <- as.data.frame(temp)
  temp$support <- paste0( '.', as.character(temp$support))
  cvResults <- cbind(cvResults, temp)

  cvResults <- cvResults[order(cvResults$model),]
  cvResults$name <- rep(name, nrow(cvResults))

  ind <- length(strsplit(as.character(cvResults$model[1]), '\\/')[[1]])
  temp <- strsplit(as.character(cvResults$model),'\\/')
  cvResults$model <- unlist(lapply(temp, '[[', ind))
  cat('duplicates?: ', which(duplicated((cvResults))) )
  return(cvResults)
}

getCVByMonth <- function(cvResults, month, metricColName) {
  cv <- cvResults[cvResults$month==month,]
  cv <- cv[order(-cv[,metricColName]),]
  cv$model <- factor(cv$model, levels=unique(cv$model)) # order levels by decreasing ROC
  cv$name <- as.factor(cv$name)
  cv$group <- paste0(cv$tType, cv$name)
  cv$tType <- as.factor(cv$tType)
  cv$tType <- factor(cv$tType, levels=levels(cv$tType)[c(3,1,2,4)])
  cv
}

rankModels <- function(cvsResults, months, topN, metricColName) {
  r <- lapply(months, function(m) {
    models <- unique(cvsResults$model)
    ranked <- lapply(models, function(m) {
      results <- max(cvsResults[cvsResults$model==m,metricColName])
    })
    names(ranked) <- models
    ranked <- unlist(ranked)
    ranked <- ranked[order(-ranked)]
    names(ranked)[1:topN]
  })
  r <- unlist(r)
  r <- r[!duplicated(r)]
  return(r)
}
```

ROC and PR plotting functions:

```r
# performance plot constants
perfTheme <- theme(legend.text=element_text(size=8, color="grey30"),
                   legend.title=element_text(size=8),
                   legend.text.align =1,
                   legend.position='right',
                   legend.key.size=unit(.8,'line'),
                   legend.margin=margin(l = -0.15, unit='cm'),
                   legend.background = element_rect(fill = "transparent", colour = "transparent"),
                   axis.text=element_text(size=10, color="#666666"),
                   axis.title.y = element_text(margin=margin(0,5,0,0),size=8, angle=0, vjust=0.5),
                   axis.title.x = element_text(margin=margin(5,0,-5,0),size=8),
                   panel.spacing = unit(.35, "lines"),
                   strip.text.x = element_text(size=9,margin = margin(.05,0,.05,0, "cm")))
scaleX <- scale_x_continuous(limits=c(0,1),breaks=seq(0,1,0.2))
scaleY <- scale_y_continuous(limits=c(0,1),breaks=seq(0,1,0.2))
perfGuide <- guides(colour = guide_legend(override.aes = list(size=3)))

# see https://learnr.wordpress.com/2009/04/29/ggplot2-labelling-data-series-and-adding-a-data-table

vplayout <- function(...) {
  grid.newpage()
  pushViewport(viewport(layout = Layout))
}
subplot <- function(x, y) viewport(layout.pos.row = x,
                                   layout.pos.col = y)
mmplot <- function(a, b) {
  vplayout()
  print(a, vp = subplot(1, 1))
  print(b, vp = subplot(2, 1))
}
# end
Layout <- grid.layout(nrow = 2, ncol = 1, heights = unit(c(2,0.5), c("null", "null")))
pd <- position_dodge(0.6) # move them .05 to the left and right

# ROC
plotCVResults <- function(cv, month, filename) {
  g <- ggplot(cv, aes(x=model, y=rocAUC, colour=tType, line=tType, group=group) ) +
    geom_point() + geom_line() +
    scale_y_continuous('Average\nArea Under the ROC Curve\n',limits=c(0.4,0.7),breaks=seq(0.4,0.7,
    theme_linedraw() +
    ggtitle(month) +
    theme(axis.text.x=element_blank(),
          axis.title.x=element_blank(),
          axis.text=element_text(size=8),
          axis.title.y = element_text(angle = 90, vjust = .5, size=8, margin=margin(0,10,0,0)),
          legend.text=element_text(size=8, color='grey30'),
          legend.title=element_text(size=8),
          legend.position = c(0.7,0.8),
          legend.key.size=unit(.8,'line'),
          plot.title = element_text(size=8, hjust=0.5)) +
    labs(color='tumor volume type') +
    scale_color_manual(values = wes_palette("Darjeeling1")[-4] )
```

```
  table <- cv[!(duplicated(cv$model)),]
  table <- table[, c('model','size','length','gap','support')]
  table <- melt(table, id='model')

  t <- ggplot(table, aes(x=model,y=variable, label=format(value))) +
    geom_text(size = 4.2, color='grey30', hjust=0.6) + theme_bw() + scale_y_discrete('cSPADE\nparam
    theme(panel.grid.major = element_blank(), legend.position = "none",
          panel.border = element_blank(),
          axis.text.x = element_blank(),
          axis.ticks = element_blank(),
          axis.text=element_text(size=8),
          axis.title.y = element_text(angle = 90, vjust = .5, size=8, margin=margin(0,11,0,5)) ) +
    theme(plot.margin = unit(c(0,.19, 0, 0), "lines")) + xlab(NULL) + ylab(NULL)

  a <- mmplot(g, t)
}

# PR
plotCVResults_PR <- function(cv, month, filename, lp) {
  g <- ggplot(cv, aes(x=model, y=prAUC, colour=tType, line=tType, group=group) ) +
    geom_point() + geom_line() +
    scale_y_continuous('Average\nArea Under the PR Curve\n',limits=c(0,0.2),breaks=seq(0,0.2,.05),
    theme_linedraw() +
    ggtitle(month) +
    theme(axis.text.x=element_blank(),
          axis.title.x=element_blank(),
          axis.text=element_text(size=8),
          axis.title.y = element_text(angle = 90, vjust = .5, size=8, margin=margin(0,13,0,0)),
          legend.text=element_text(size=8, color='grey30'),
          legend.title=element_text(size=8),
          legend.position=lp,
          legend.key.size=unit(.8,'line'),
          plot.title = element_text(size=8, hjust=0.5)) +
    labs(color='tumor volume type') +
    scale_color_manual(values = wes_palette("Darjeeling1")[-4] )

  table <- cv[!(duplicated(cv$model)),]
  table <- table[, c('model','size','length','gap','support')]
  table <- melt(table, id='model')

  t <- ggplot(table, aes(x=model,y=variable, label=format(value))) +
    geom_text(size = 4.2, color='grey30', hjust=0.6) + theme_bw() + scale_y_discrete('cSPADE\nparam
    theme(panel.grid.major = element_blank(), legend.position = "none",
          panel.border = element_blank(),
          axis.text.x = element_blank(),
          axis.ticks = element_blank(),
          axis.text=element_text(8),
          axis.title.y = element_text(angle = 90, vjust = .5, size=8, margin=margin(0,11,0,5)) ) +
    theme(plot.margin = unit(c(0,.19, 0, 0), "lines")) + xlab(NULL) + ylab(NULL)

  a <- mmplot(g, t)
}
```

4. **Parse the cross-validation log files**

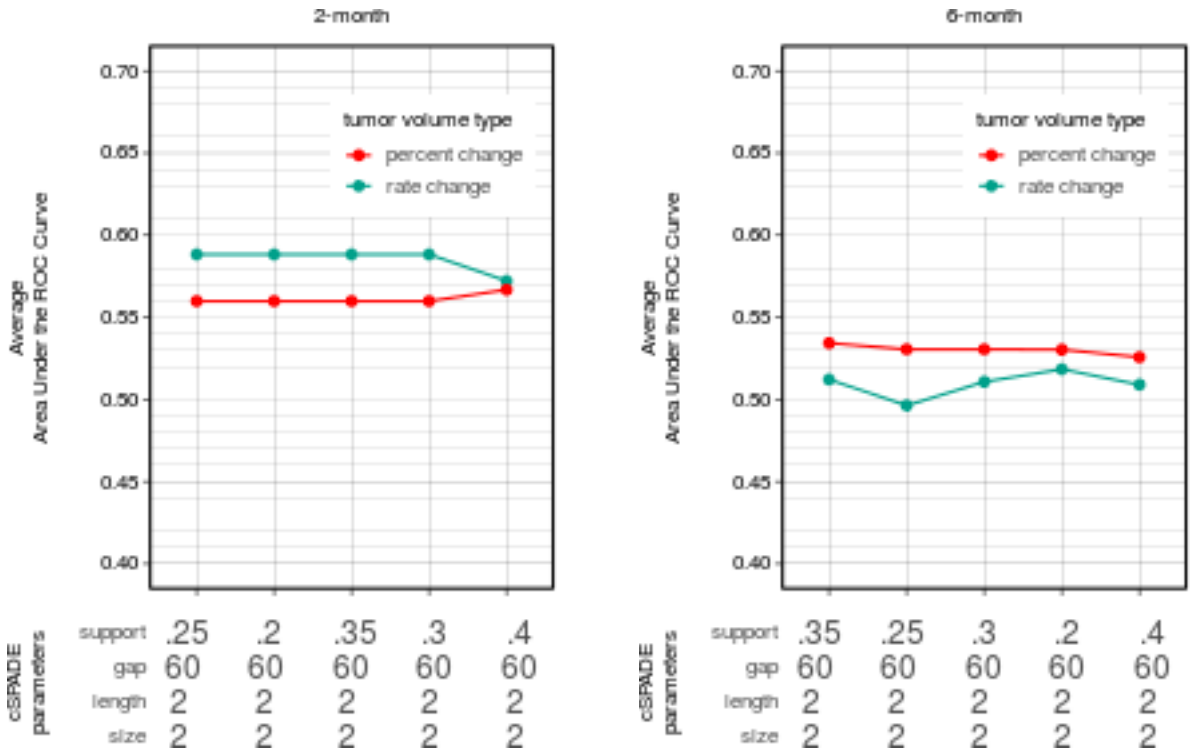
The following assumes a specific location and format of the log file (e.g., the rows skipped while reading `.log` files)

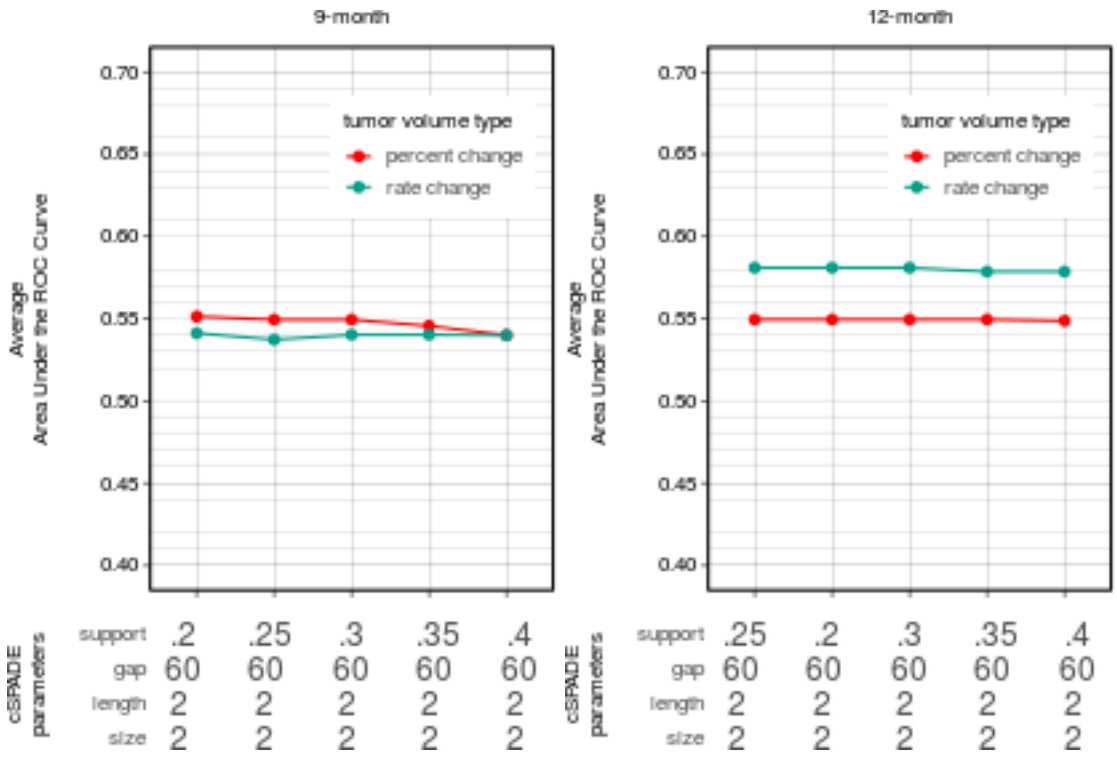
```
dataDir <- '~/gbm_spm_example_multi/logits'
theseDirs <- list.dirs(path=dataDir, full.names=F, recursive=F)
theseDirs
#> [1] "sup0.25g60l2z2" "sup0.2g60l2z2"  "sup0.35g60l2z2" "sup0.3g60l2z2"
#> [5] "sup0.4g60l2z2"

# types and typenames are matched
types <- c('percent','rateChange') # tumor volume types expected in log filename
typenames <- c('percent change','rate change') # corresponding pretty names of tumor volume types
months <- c('2-month','6-month','9-month', '12-month') # month names for classification labels

cvs <- lapply(theseDirs, function(f) {
  cat(f, '\n')
  logFiles <- list.files(path=file.path(dataDir,f), pattern='\\.log$') # locate log files

  # extract results for each tumor volume type
  results <- lapply(seq(1,length(types)), function(i) {
    n <- logFiles[grepl(types[i],logFiles)] # read in a log file
    if (length(n) > 0) {
      log <- file.path(dataDir,f,n)
      con <- file(log)
      open(con)
      resultsCols <- read.table(con, skip=114, nrow=1, stringsAsFactors = F)  # expected results co
      r2 <- read.table(con, skip=0, nrow=1) # expected location of CV results for each classificat
      r6 <- read.table(con, skip=25, nrow=1)
      r9 <- read.table(con, skip=25, nrow=1)
      r12 <- read.table(con, skip=25, nrow=1)
      close(con)

      r <- do.call('rbind', list(r2,r6,r9,r12)) # combine and format
      r <- r[,-1]
      colnames(r) <- c(resultsCols[1,])

      r$month <- months # attach additional info
      r$tType <- rep(typenames[i], nrow(r))
      r$model <- rep(f, nrow(r))
      temp <- do.call(rbind, lapply(r$model, function(n) regmatches(n, gregexpr('[0-9]+',n))[[1]][2
      colnames(temp) <- c('support', 'gap', 'length', 'size')
      temp <- as.data.frame(temp)
      temp$support <- paste0( '.', as.character(temp$support))
      r <- cbind(r, temp)
      r$name <- 'without'
      r
    } else {
      cat('\t ... file not found for ', typenames[i] , '\n')
    }
  })
  results <- do.call('rbind',results)
})
#> sup0.25g60l2z2
#> sup0.2g60l2z2
```

```
#> sup0.35g60l2z2
#> sup0.3g60l2z2
#> sup0.4g60l2z2
cvs <- do.call('rbind',cvs)
cat('duplicates?: ', which(duplicated((cvs))) )
#> duplicates?:
cvs[1:5,]
#>        prAUC    rocAUC     prAUCSD   rocAUCSD     lambda    month
#> 1 0.04622387 0.5598533 0.013537090 0.03944685 0.05437176  2-month
#> 2 0.06864538 0.5304702 0.011415910 0.07776483 0.03414705  6-month
#> 3 0.10674390 0.5494308 0.021065460 0.01920351 0.03111352  9-month
#> 4 0.13006210 0.5494858 0.089507720 0.04368807 0.09501620 12-month
#> 5 0.01244991 0.5882589 0.006009641 0.06704849 0.20000000  2-month
#>             tType          model support gap length size    name
#> 1 percent change sup0.25g60l2z2     .25  60      2    2 without
#> 2 percent change sup0.25g60l2z2     .25  60      2    2 without
#> 3 percent change sup0.25g60l2z2     .25  60      2    2 without
#> 4 percent change sup0.25g60l2z2     .25  60      2    2 without
#> 5    rate change sup0.25g60l2z2     .25  60      2    2 without
```

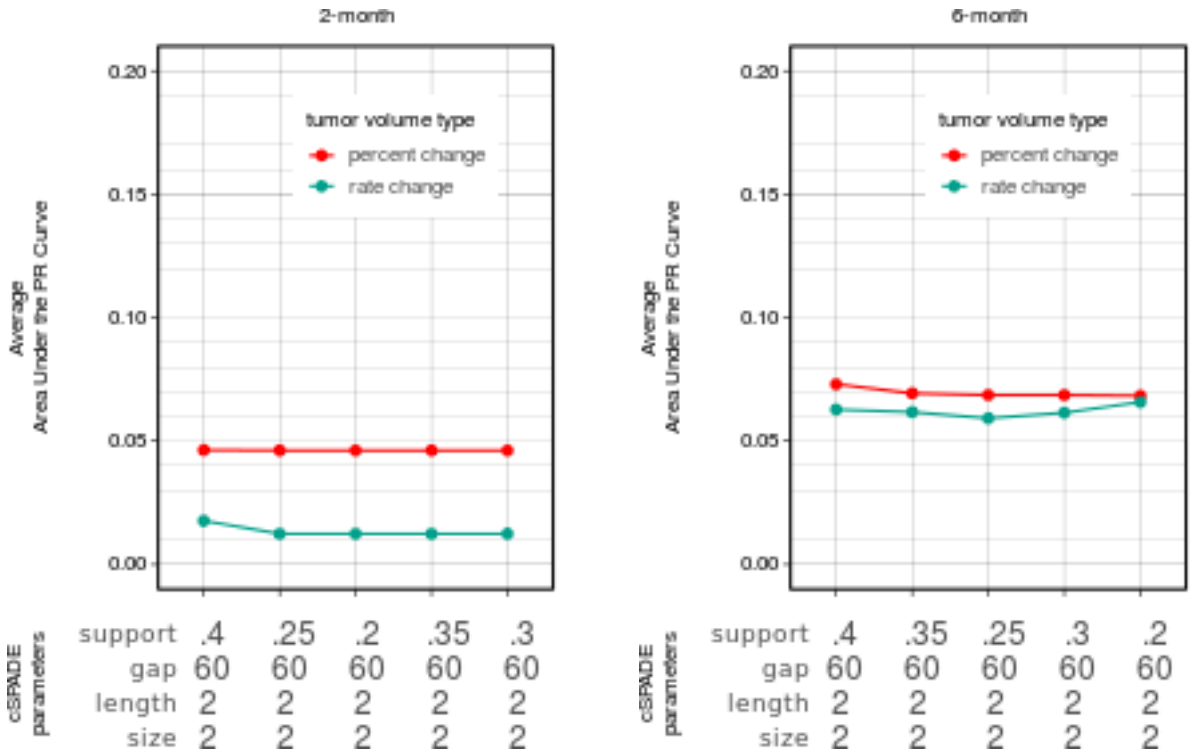
5. **Plot cross-validation results**

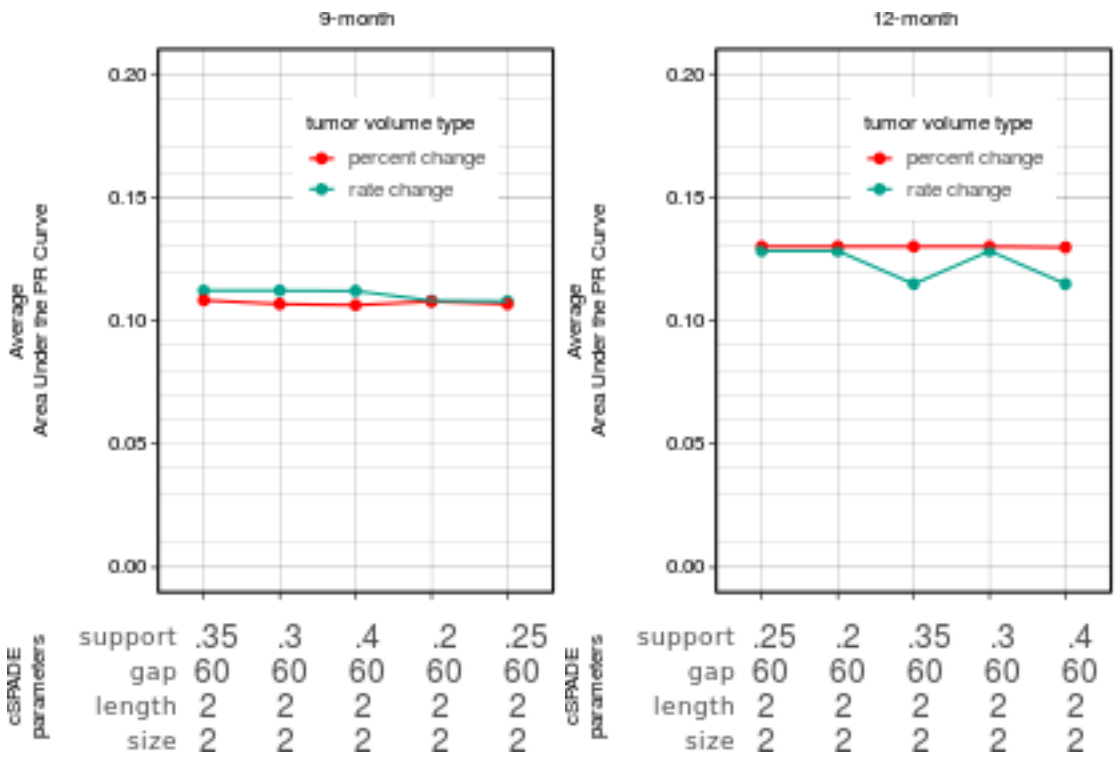
```
# individual plots
spmROC <- lapply(months, function(m) plotCVResults(getCVByMonth(cvResults=cvs, m, 'rocAUC'),
                                                   m,
                                                   '' ) )
```






```
spmPR <- lapply(months, function(m) plotCVResults_PR(getCVByMonth(cvResults=cvs, m, 'prAUC'),
                                                      m,
                                                      '',
                                                      c(.6,.8)) )
```





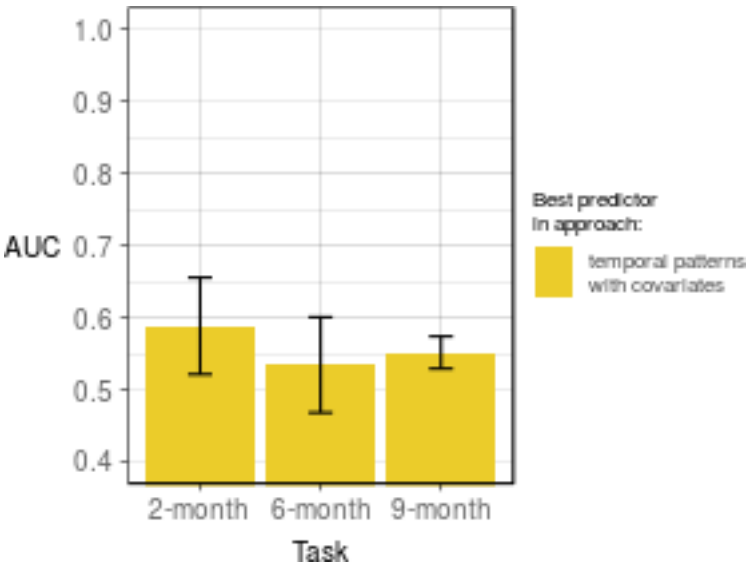
6. **Identify best model parameters from cross-validation**

```
# cv logit results
bestLogitSpm <- lapply(months, function(m) {
  z <- cvs[which(cvs$month==m),]
  b <- z[which.max(z$rocAUC),]
  b <- b[,colnames(b) %in% c('rocAUC','rocAUCSD','tType','month','model', 'gap','length','name')]
  rownames(b) <- b$model
  b
})
bestLogitSpm <- do.call('rbind', bestLogitSpm)
colnames(bestLogitSpm) <- c('rocAUC','rocAUCSD','month','name','model','maxgap','maxlength')
bestLogitSpm$label <- getClassLabels() # assumes row order already matches
bestLogitSpm$fitname <- c('fit2m','fit6m','fit9m','fit12m')
bestLogitSpm
#>                     rocAUC   rocAUCSD    month           name
#> sup0.25g60l2z2  0.5882589 0.06704849  2-month    rate change
#> sup0.35g60l2z2  0.5342622 0.06626262  6-month percent change
#> sup0.2g60l2z2   0.5513217 0.02236023  9-month percent change
#> sup0.25g60l2z21 0.5810034 0.01947099 12-month    rate change
#>                           model maxgap maxlength      NA         label
#> sup0.25g60l2z2  sup0.25g60l2z2     60         2 without  survivalIn60
#> sup0.35g60l2z2  sup0.35g60l2z2     60         2 without survivalIn180
#> sup0.2g60l2z2    sup0.2g60l2z2     60         2 without survivalIn270
#> sup0.25g60l2z21 sup0.25g60l2z2     60         2 without survivalIn360
#>                 fitname
#> sup0.25g60l2z2    fit2m
#> sup0.35g60l2z2    fit6m
#> sup0.2g60l2z2     fit9m
#> sup0.25g60l2z21  fit12m
```

7. **Plot the cross-valiation results**

There is only one approach explored in this example, i.e., approach 3, using temporal patterns and patient covariates to predict residual survival. The paper explores 2 other approaches.

```
comp <- bestLogitSpm[,c('rocAUC','rocAUCSD','month','name')]
comp$lower <- comp$rocAUC - comp$rocAUCSD
comp$higher <- comp$rocAUC + comp$rocAUCSD
comp$approach <- 3
comp <- do.call('rbind', list(comp))
comp$approach <- as.factor(comp$approach)
comp$approach <- factor(comp$approach, levels=levels(comp$approach),
                        labels=c( 'temporal patterns\nwith covariates'))
pd <- position_dodge(0.6) # move them .05 to the left and right

comparedPlot <- ggplot(comp[!(comp$month=='12-month'),], aes(x=month, y=rocAUC, fill=factor(approac
  geom_bar(stat='identity',position=position_dodge()) +
  scale_y_continuous(limits = c(.4,1), oob=rescale_none, breaks = seq(.4,1,0.1)) +
  theme_linedraw() +
  # ggtitle('') +
  perfTheme +
  geom_errorbar(aes(ymin=lower, ymax=higher), width=.2, position=position_dodge(.9)) +
  labs(fill='Best predictor\nin approach:') + xlab('Task') + ylab('AUC') +
  scale_fill_manual(values = wes_palette("Zissou1")[3:5] ) +
  theme(legend.text.align=0,
        legend.key.height=unit(1.5,'line'),
        legend.text = element_text(size=8),
        legend.title = element_text(size=8),
        axis.text=element_text(size=10),
        axis.title.y=element_text(size=10),
        axis.title.x=element_text(size=10)) +
  guides(fill = guide_legend(override.aes = list(size=6)))
print(comparedPlot)
```



8. **Retrain and apply to test set**

A model is selected, retrained on the training partition, and applied to the testing partition for each prediction task. For the sake of this example, let's assume rate change as the tumor volume measurement

was the best performer across all tasks.

Prep for retraining:

```
seed <- 9
metric <- 'rocAUC'
expDir <- dirname(dataDir)

# data partitions
dataPartitions <- file.path(expDir, 'train_test_partitions.rds')
partitions <- readRDS(file=dataPartitions)

# unwanted features and labels that should not be in training data

needToRemove <- c('id','iois','eventID', # remove ids
                  bestLogitSpm$label, # remove labels
                  'IDH1') # not interested

# where to find the 'rate change' data for training
selectedTType <- 'rate'
selectedTTypeFN_spm <- 'featureVectors_rateChange.rds'
selectedTTypeFN_cvLogit <- 'logits_CVresults_rateChange.rds'
```

Get model performance results in retraining and testing:

```
retrainResults <- lapply(seq(1, nrow(bestLogitSpm)), function(i) {
  selected <- bestLogitSpm[i,]

  cat('\n....', selected$label ,' ...\n')

  # cv results
  cvLogitPath <- file.path(expDir,'logits', selected$model, selectedTTypeFN_cvLogit)
  cvResults <- readRDS(cvLogitPath)
  cvResults <- cvResults[[selected$fitname]]

  # get lambda from best cv results
  cvResults <- cvResults$avg.CVPerformance[which.max(cvResults$avg.CVPerformance$rocAUC),]
  cat('\n\n','Summary to repeated CV results (lambdas for retraining) ...\n\n')
  print(cvResults)


  # get data
  dataPath <- file.path(expDir,'spm', selected$model, selectedTTypeFN_spm)
  data <- readRDS(dataPath) # features and labels for each clinical visit
  names <- colnames(data)
  data <- data.frame(id=row.names(data),data)
  colnames(data) <- c('id',names)
  cat('... \n overall samples: ', nrow(data), '\n')

  data <- prepLaterality(data)
  data <- prepLocation(data)

  cat('... \n removing clinical visits with no history of length maxgap:', selected$maxgap, '\n')
  data <- removeVisits(data = data,
                       maxgap = as.numeric(as.character(selected$maxgap)),
                       maxlength = as.numeric(as.character(selected$maxlength)),
```

```
                        tType = selectedTType,
                        save=F,
                        outDir=outputDir)
  start.local <- Sys.time()
  cat('\n....', cvResults$lambda ,' ...\n')
  label <- selected$label

  fit <- retrainLogit(data = data[data$id %in% partitions$train,],  #training data
                      formula = as.formula(paste0(selected$label,'~.')),
                      labelName = selected$label,
                      lambda = cvResults$lambda,
                      lasso=TRUE, # using lamba
                      needToRemove=needToRemove,
                      createModelMatrix=FALSE,
                      metric=metric,
                      verbose=TRUE,
                      seed=seed)
  cat('\n'); print(Sys.time() - start.local)

  trainResults <-  getTrainResults(fit)
  testResults <- getTestResults(fit = fit,
                                data = data[data$id %in% partitions$test,], #testing data
                                labelName = selected$label,
                                formula = as.formula(paste0(selected$label,'~.')),
                                needToRemove=needToRemove,
                                createModelMatrix=FALSE)

  testResults$selectedThreshold <-  trainResults$selectedThreshold # use the threshold selected fr

  list(fit=fit, train=trainResults, test=testResults)
})
#>
#> .... survivalIn60  ...
#>
#>
#>  Summary to repeated CV results (lambdas for retraining) ...
#>
#>          prAUC    rocAUC     prAUCSD   rocAUCSD lambda
#> 100 0.01244991 0.5882589 0.006009641 0.06704849    0.2
#> ...
#>  overall samples:  1500
#> ...
#>  removing clinical visits with no history of length maxgap: 1
#> ...visits left for training:  1340
#>
#> .... 0.2  ...
#> Fitting alpha = 1, lambda = 0.2 on full training set
#>
#> Time difference of 0.6109157 secs
#>
#> .... survivalIn180  ...
#>
#>
```

```
#>  Summary to repeated CV results (lambdas for retraining) ...
#>
#>         prAUC    rocAUC    prAUCSD  rocAUCSD     lambda
#> 82 0.06175422 0.5121156 0.01104318 0.0866874 0.03747635
#> ...
#>  overall samples:  1500
#> ...
#>  removing clinical visits with no history of length maxgap: 1
#> ...visits left for training:  1340
#>
#> .... 0.03747635  ...
#> Fitting alpha = 1, lambda = 0.0375 on full training set
#>
#> Time difference of 0.3991973 secs
#>
#> .... survivalIn270  ...
#>
#>
#>  Summary to repeated CV results (lambdas for retraining) ...
#>
#>         prAUC    rocAUC    prAUCSD   rocAUCSD     lambda
#> 89 0.1081031 0.5411896 0.02410388 0.04195756 0.07187627
#> ...
#>  overall samples:  1500
#> ...
#>  removing clinical visits with no history of length maxgap: 1
#> ...visits left for training:  1340
#>
#> .... 0.07187627  ...
#> Fitting alpha = 1, lambda = 0.0719 on full training set
#>
#> Time difference of 0.4219236 secs
#>
#> .... survivalIn360  ...
#>
#>
#>  Summary to repeated CV results (lambdas for retraining) ...
#>
#>         prAUC    rocAUC    prAUCSD   rocAUCSD    lambda
#> 95 0.1282788 0.5810034 0.06671858 0.01947099 0.1256058
#> ...
#>  overall samples:  1500
#> ...
#>  removing clinical visits with no history of length maxgap: 1
#> ...visits left for training:  1340
#>
#> .... 0.1256058  ...
#> Fitting alpha = 1, lambda = 0.126 on full training set
#>
#> Time difference of 0.3907464 secs
names(retrainResults) <- bestLogitSpm$fitname
```

Contents of retrainResults:

```
names(retrainResults) # for each model
#> [1] "fit2m"  "fit6m"  "fit9m"  "fit12m"
names(retrainResults$fit2m) # for each prediction
#> [1] "fit"   "train" "test"
names(retrainResults$fit2m$train) # results in train
#> [1] "preds"             "performance"       "selectedThreshold"
names(retrainResults$fit2m$test) # results in test
#> [1] "preds"             "performance"       "data"
#> [4] "selectedThreshold"
```

If we were to chose a threshold for classifiying 2-month residual survival, the results would be:

```
retrainResults$fit2m$train$selectedThreshold #train
#>           selectROCThreshold(trainPred)
#> cutoff                        0.6313534
#> fp                           73.0000000
#> tp                           10.0000000
#> tn                          942.0000000
#> fn                           12.0000000
#> spec                          0.9280788
#> sens                          0.4545455
#> prec                          0.1204819
#> recall                        0.4545455
#> f1score                       0.1904762
#> accuracy                      0.9180328
#> cutoffInd                     2.0000000
retrainResults$fit2m$test$selectedThreshold #test
#>           selectROCThreshold(trainPred)
#> cutoff                        0.6313534
#> fp                           73.0000000
#> tp                           10.0000000
#> tn                          942.0000000
#> fn                           12.0000000
#> spec                          0.9280788
#> sens                          0.4545455
#> prec                          0.1204819
#> recall                        0.4545455
#> f1score                       0.1904762
#> accuracy                      0.9180328
#> cutoffInd                     2.0000000
```

9. **Plot final model's performance results in retraining and testing**

Extract and structure the performance results:

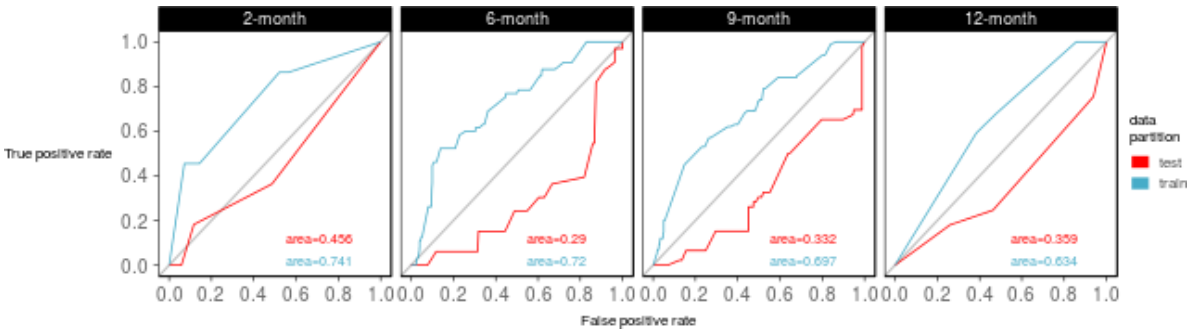
```
rocsList <- lapply(retrainResults, function(fit) {list(train=fit$train$performance$roc,
                                                       test =fit$test$performance$roc )})
aucsList <- lapply(retrainResults, function(fit) {list(train=fit$train$performance$aucROC,
                                                       test =fit$test$performance$aucROC )})
prsList <- lapply(retrainResults, function(fit) {prs <- list(train=fit$train$performance$pr,
                                                             test =fit$test$performance$pr )})
praucsList <- lapply(retrainResults, function(fit) {list(train=fit$train$performance$aucPR,
                                                         test =fit$test$performance$aucPR)})
names(rocsList) <- months # assumes months and order of retrainResults are the same
names(aucsList) <- months
```

```
names(prsList) <- months
names(praucsList) <- months
```

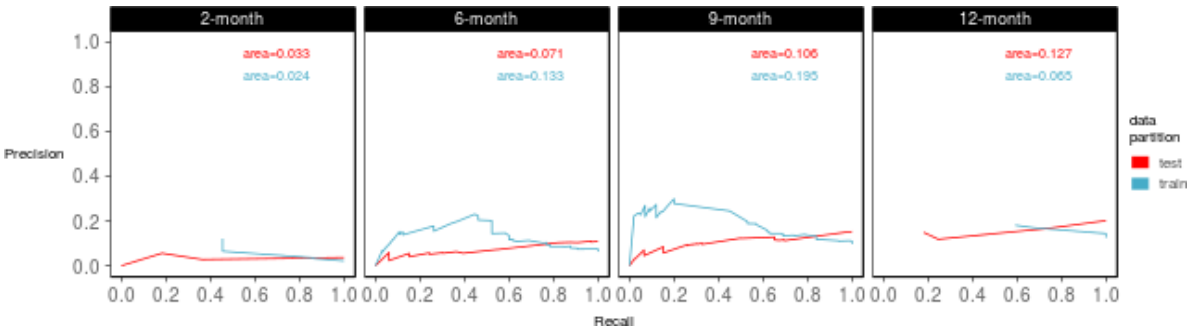
Plot ROC and PR of final model's test and train results

```
scaleColor <- scale_color_manual(values = c(wes_palette("Darjeeling1")[1], wes_palette('FantasticFo
lo <- c(2,3,4,1)

finalModelROC <- plotMultiROC(fn = '',
                              rocsList = rocsList[1:4],
                              aucsList = aucsList[1:4],
                              listOrder = lo,
                              outDir = NULL,
                              legendTitle = 'data\npartition',
                              printIt=F,
                              perfTheme = perfTheme,
                              scaleX = scaleX,
                              scaleY = scaleY,
                              perfGuide = perfGuide) +
  theme(legend.text.align=0) +
  scaleColor
finalModelPR <- plotMultiPR(fn = '',
                            prsList = prsList[1:4],
                            aucsList = praucsList[1:4],
                            legendTitle = 'data\npartition',
                            outDir = NULL,
                            listOrder = lo,
                            printIt=F,
                            perfTheme = perfTheme,
                            scaleX = scaleX,
                            scaleY = scaleY,
                            perfGuide = perfGuide) +
  theme(legend.text.align=0) +
  scaleColor
```

```
print(finalModelROC)
```




```
print(finalModelPR)
```


2-month
6-month
9-month
12-month
area=0.033
area=0.024
area=0.071
area=0.133
area=0.106
area=0.195
area=0.127
area=0.065
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
data
partition
test
train